GRAPHIC USER INTERFACE AND METHOD FOR CREATING SLIDE SHOWS

CROSS REFERENCE TO RELATED APPLICATION

[001] The present application is a continuation-in-part of U.S. patent application serial no. 10/053,075, filed January 18, 2002, which is a continuation-in-part of U.S. patent application serial no. 09/880,397, filed June 12, 2001, which in turn is a continuation-in-part of U.S. patent application serial no. 09/785,049, filed Feb. 15, 2001, for which priority is claimed. The entireties of the prior applications are incorporated herein by reference.

FIELD OF THE INVENTION

[002] The invention relates generally to graphic user interfaces (GUIs), and more particularly to a graphic user interface (GUI) and method for creating media presentation.

BACKGROUND OF THE INVENTION

[003] The widespread use of digital still and video cameras has increased demands for computer programs to manage and/or edit electronic media, as well as create sequential media presentations, such as picture slide shows. Currently, there are a number of sophisticated computer programs that allow users to create sequential media presentation.

[004] However, for an average computer user, most of these computer programs present challenges in the form of complexity and unfamiliarity in using these computer programs to create sequential media presentations. As an example, a user may need to learn and perform a number of steps using pull-down menus in an unfamiliar computer environment to create a simple picture slide show.

[005] In view of this concern, what is needed is a graphic user interface and method for readily creating sequential media presentations.

SUMMARY OF THE INVENTION

[006] A graphic user interface (GUI) and method for readily creating and manipulating sequential media presentations, such as pictures slide shows, allows a user to select media for a sequential media presentation in the order in which the selected media is to be displayed in the presentation by simply drawing an arrow that intersect onscreen representations of the selected media. In addition, the user may manipulate various parameters of the selected media as the selected media are being played back.

[007] Other aspects and advantages of the present invention will become apparent from the following detailed description, taken in conjunction with the accompanying drawings, illustrated by way of example of the principles of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[008] Figure 1 depicts the calling forth of an arrow logic using a color from an inkwell, an arrow switch and the action of drawing onscreen.

[009] Figure 2 depicts the creation of a slide show with the drawing of an arrow onscreen.

[010] Figure 3 shows one result from the drawing of the blue arrow 3 in Figure 2, namely, the creation of a Slide Show VDACC.

[011] Figure 4 depicts a method of playing a slide show.

[012] Figure 5 depicts changing the onscreen time for an image in a slide show by typing a new number into its duration editor.

[013] Figure 6 depicts using a fader to alter the onscreen time for one or more images in a slide show.

[014] Figure 7 illustrates changing the order of the images in a slide show by dragging an image from one location to another within the Slide Show VDACC.

[015] Figure 8a depicts inserting an image into a slide show by the drawing of an arrow.

[016] Figure 8b shows the newly created list of images in the Slide Show resulting from the insert made in Figure 8a.

[017] Figure 9 is an illustration of a Play Rectangle (PR).

[018] Figure 10 depicts playing slide show images in a Play Rectangle (PR).

[019] Figure 11a depicts resizing a Play Rectangle (PR) and the image playing within it.

[020] Figure 11b depicts moving a Play Rectangle (PR) and the image playing within it.

[021] Figure 11c is a description of an image being force fit to a PR and thereby skewed in the process.

[022] Figure 12 depicts the general use of "Fit Image to PR."

[023] Figure 13 show selecting "Fit PR to Image" and its affect on an image playing in a Play Rectangle (PR).

[024] Figure 14 shows the default alignment of images' upper left corner to the upper left corner of a PR as they are played back within a PR in a slide show.

[025] Figure 15 shows the alignment of images' center point to the center point of a PR as they are played back within a PR in a slide show.

[026] Figure 16 refers to the entry Square %" and its use with a PR in a slide show.

[027] Figure 17 illustrates how the definition of a square is calculated from the entry "% of landscape width."

[028] Figure 18 is a flow chart for playing a slide show.

[029] Figure 19 is a flow chart for showing a slide in a Play Rectangle (PR).

[030] Figure 20 is a flow chart for hiding a slide in a slide show.

[031] Figure 21 is a flow chart for using a Play Rectangle.

[032] Figure 22 is a flow chart for changing a Play Rectangle geometry.

DETAILED DESCRIPTION

[033] A graphic user interface (GUI) in accordance with an embodiment of the invention permits users to create sequential media without being required to use frames and the placement of slides as utilized by presentation software common in the art. Instead of selecting commands from pull down menus or the like, this invention supports the creation of sequential media by graphical means, which does not utilize pull down menus, task bars or the like.

[034] To create a slide show with the software of this invention, a user selects a color from a graphic inkwell and then draws an arrow which has an arrow logic assigned to it. This arrow logic is recalled by selecting a color that corresponds to a particular arrow logic. See pending U.S. patent application serial no. 09/880,397. One of the key benefits of the invention is that it permits a user to draw a single arrow to select multiple items, arrange them in a sequential order and have a duration assigned to each item.

[035] Referring to Figure 1, a user selects a color 1 in an inkwell 2. Then an arrow switch 4 is turned on and an arrow 3 is drawn that calls forth a specific arrow logic 5. The default color for the arrow logic "sequential ordering" is the color blue. This color is a user modifiable parameter. In other words, it can be changed to another color by the user. This is done according to the approaches described in the pending U.S. patent application serial no. 09/880,397. One way to draw an arrow is to first turn on the Arrow switch 4 and then left-click on a global drawing surface, referred to herein as "Blackspace", and draw an arrow 3.

[036] In Figure 2, a blue arrow 3 is drawn to intersect various picture files in a Picture Files List 6, which is shown as a Picture Files VDACC 7. For information about VDACCs, see U.S. patent application entitled "Intuitive Graphic User Interface with Universal tools", which is being simultaneously filed and specifically incorporated by reference herein.

[037] The picture files are represented by text objects in this list 6. Each text object that the shaft of the arrow 3 intersects will be added to the source list 8 of the

arrow. Any object intersected by the head (tip) of the arrow 3 is added to the target list 9 of the arrow. In this illustration, no object is intersected and the target list 9 is empty. The shaft of the arrow 3 is a line that must intersect some part of one or more characters belonging to a text object in the list 6 in order for that text object to be selected by the drawing of the arrow 3. In Figure 2 non-contiguous text objects in the list have been selected by the drawing of the arrow 3. A user can literally draw that line of the arrow in any direction to intersect any one or more text objects in a list and this will result in the selection of all items that have had any part of one or more of their characters intersected by the arrow.

[038] The selected objects appear in the source list 8 of the arrow 3. In reality, they are listed according to their IDs, not their text object name. For the sake of illustration, the selected objects are shown with both in the source list 8 of the arrow 3. The blue arrow 3 is drawn such that the tip of the arrow points to blank Blackspace. In this case, the target list of the arrow will be empty, indicating that a fresh Slide show is to be created rather than updating an existing one. After drawing the blue arrow 3 and upon the mouse up-click, the arrow's arrowhead 10 turns white. This indicates that the arrow has been properly drawn.

[039] Figure 3 shows the result of the drawing of the blue arrow 3 in Figure 2. When the white arrowhead of the blue arrow 3 is left-clicked on (or its equivalent), the blue arrow disappears along with its white arrowhead and the Slide Show VDACC 11 appears. Inside this VDACC are a number of smaller VDACCs that have one image inside of each of them. These images are presented, each in their own small VDACC 14, in the order that they were intersected by the drawn blue arrow, as shown in Figure 2. Above each picture is a duration editor 13. This is shown in seconds -- 5.000. A feature of the duration editor is also shown, namely, that for the slide that is currently playing, its duration editor turns green 12. Also shown in Figure 3 is a play cursor 15 which appears at the far left edge of the first image in the first small VDACC 14 in the Slide Show VDACC 11.

[040] Referring to Figure 4, to play a slide show in a Slide Show VDACC 11, the DM Play switch 16 is turned on and the green play cursor 15 starts to move

through the first image in the Slide Show VDACC 11. The length of time that it takes for the play cursor to move through each slide equals the length set by each slide's duration editor. In the case of this example each image is set to a duration of 5.000 seconds.

[041] The Slide Show VDACC is not generally used for playing back a slide show, because the images are too small. The value of the Slide Show VDACC is as an overview and for editing the slide show. Three types of edits will be discussed.

[042] 1. Changing the onscreen time of one or more slide show images.

[043] 2. Changing the order of one or more slide show images.

[044] 3. Inserting one or more images into a slide show.

1. Changing the onscreen time of one or more slide show images.

[045] There are many ways to change the onscreen time of one or more slide show images. One of these ways is shown in Figure 5. Turn on the text switch 17 and left-click (or its equivalent) to place a text cursor into a duration editor. Then type new text into the duration editor. This is shown as duration editor 18, which has been changed to 2.000 seconds. This means that when this image plays in the slide show it will remain onscreen for only 2 seconds, not 5 seconds.

[046] Another way to change the onscreen time of images in a slide show is shown in Figure 6. Here a RDraw switch 22 has been turned on to activate the Draw Mode. Then a fader 19 is drawn and a red arrow 20 with a "control" arrow logic assigned to it is drawn to intersect the fader and point to the Slide Show VDACC 11. This control arrow logic can be stated by the following sentence: "the object(s) that the arrow is drawn from controls the object(s) that the arrow points to." After drawing the arrow 20 as shown in Figure 6 its arrowhead turns white. Clicking on the white arrow establishes the fader as a duration control for all of the images in the Slide Show VDACC at one time.

[047] If the fader 19 is then moved to the right, the duration values for all of the images in the Slide Show VDACC 11 are increased. If the fader is moved to the left, these duration values are decreased.

[048] As an alternate, if the red arrow is drawn to intersect the fader 19 and only one picture in the Slide Show VDACC 11, then this fader will control the duration time for that image and no others. The operation of the fader operation is the same as what has been just described.

[049] There are various ways to ensure that a drawn fader is programmed to be a duration control. One way is that the software has a default setting determining that any fader intersected by a red arrow, where the arrow points to a Slide Show VDACC or to any image in a Slide Show VDACC, will automatically be programmed to be a duration fader.

[050] Another way to ensure that a drawn fader is programmed to be a duration fader is illustrated in Figure 6. A text object 5000ms 21 was typed onscreen. This text object 21 could be any time value, like seconds, or frames, etc. The text object 21 is typed to intersect the fader 19. As an alternate, the text object 21 could be dragged to intersect the fader. In either case, when a collision is detected the text object 21 snaps to a position that is determined by the software. It could be directly above the fader 19, below it or anywhere on screen. Once this text object 21 appears, the user can drag it to a new position where it will remain until the user moves it again.

2. Changing the order of one or more slide show images.

[051] Figure 7 shows an image 23 being dragged from its previous position 24 in a slide show VDACC to a new position as the first image in the slide show. When this image 23 is dragged to overlap the first image, a blue insert rectangle 14 appears around this first image. This blue rectangle 14 tells the user which image the dragged image will be inserted in front of. Upon a mouse up-click, the dragged image 23 will be inserted in front of the first image in the Slide Show VDACC 11a. When the reordering of the slides occurs, all of the small VDACCs in the Slide Show VDACC are repositioned, as shown by the Slide Show VDACC 11b. The dragged image 23 snaps into the first slide position and what was the first image moves to position two. What was the third slide is moved to position four. Since the play cursor is sitting at

the left edge of image one, the duration editor for slide one (the newly dragged image) turns green to indicate that it is the slide currently selected to play.

3. Inserting one or more images into a slide show.

[052] In Figure 8a, a blue arrow 3 has been drawn to intersect the text object "flower 26.bmp" in the Picture Files VDACC. The tip of this arrow has been drawn to overlap the third image in the Slide Show VDACC. The source item for this arrow is the text object "flower 26" which represents a photo saved on the hard drive. The target item for this arrow is the third small VDACC 27 in the Slide Show VDACC 11.

[053] Upon the mouse up-click, the image "flower 26" snaps in front of image 27. All of the images in the Slide Show VDACC 11 that exist to the right of the newly placed image "flower 26" will move the space of one image to the right.

[054] Figure 8b shows the newly created list of images in the Slide Show VDACC 11. Image 28 has now been inserted into the Slide Show VDACC 11 in front of image 27. All of the other images occurring later than image 27 have been moved to the right in the Slide Show VDACC 11.

Playing a slide show in a Play Rectangle (PR).

[055] Figure 9 shows a Play Rectangle (PR) 29. A user can create a PR, by first creating a switch 33, e.g., by using Object Points (see pending U.S. patent application serial no. 10/103,680, which is incorporated herein by reference), and then typing the letters PR on the switch. Then when the switch 33 is activated, e.g., left-clicked on, the PR 29 immediately appears. For a detailed explanation of the PR, see U.S. patent application entitled "Method for Creating and Using Linear Timeline and Play Rectangle", which is being simultaneously filed and specifically incorporated by reference herein.

[056] In appearance, the PR 29 looks like a timeline with four sides. But the PR offers functions beyond that of any timeline. For instance, the PR can be used to both reposition and resize the images playing back as sequential media (as in a slide

show), which are sequentially ordered by selecting them with a blue arrow and pointing that arrow to Blackspace.

[057] The PR 29 has a play cursor 15 that moves along the outer edge of the PR as media plays. Floating the mouse cursor 30 over the play cursor 15 will cause a parameter 31 to appear showing the exact time of the play cursor's location along the PR 29. This play cursor 15 can be left-clicked on and dragged clockwise or counter-clockwise along the PR. As the play cursor 15 is dragged the parameter 31 changes to always show the exact location of the play cursor along the PR. As the play cursor is dragged, the media assigned to the PR 29 will play backwards or forwards, depending upon the direction and speed of the drag. A clockwise drag yields a forward direction of play and a counter-clockwise drag yields a reverse or backwards direction of play.

[058] Using the play cursor on a PR is a strong navigational tool. One reason is that when a blue arrow is drawn to create sequential media and place it into a Slide Show VDACC, bringing either a linear timeline or a rectangular timeline (a PR) onscreen and then activating the DM Play switch (or its equivalent) will automatically change the overall time of both timelines to equal the overall length of the sequential media, e.g., the slide show. For instance, if the slide show is 60 seconds long, when the DM Play switch is turned on to play the slide show, any visible timeline will have its overall length of time changed to equal 60 seconds.

[059] With the full length of a media presentation shown along a PR, a user can left-click anywhere along the PR and place the play cursor to play any section of that media. Everything is visible to the user on the PR, including the images themselves.

[060] Figure 10 shows a special feature of the PR, namely, that images in a piece of sequential media, e.g., a slide show, can be made to play such that they fill the inside area of the PR 29. To accomplish this a user creates a PR switch 33, by using Object Points to create a switch and then by typing the letters PR or their equivalent on the switch. Then the switch 33 is turned on by clicking on it. When this switch 33 is activated, the PR 29 appears onscreen at a default size set in the

timeline software. When the PR 29 appears, the image 36 that is currently set to be played in the slide show appears in the PR such that it fills the entire inner area of the PR.

[061] Then when the DM Play switch 34 or its equivalent is activated, the images play back in sequential order. This order equals the order that they appear in the Slide Show VDACC. The onscreen time for each of these images is determined by the duration editor 12 for each slide.

[062] The action of having the images of the slide show automatically appear in the PR 29 is controlled by an entry in the Info Canvas 35 for the DM Play switch 34. For information on Info Canvases, see U.S. patent application entitled "Intuitive Graphic User Interface with Universal tools", which is being simultaneously filed. To access this entry, a user right-clicks on the DM Play switch 34 and its Info Canvas 35 appears. In this Info Canvas 35 is a category entitled: "Slide Show." When this category is activated, i.e., left-clicked on, the entries that belong to this category appear onscreen. In the Info Canvas 35, three entries are selected: (1) Show Images in VDACCs, (2) Show images in Play Rectangle, and (3) Show DM Cursor in Images.

[063] The entry "Show Images in VDACCs" activates the appearance onscreen of the Slide Show VDACC 11. The entry "Show images in Play Rectangle" activates the usage of the PR 29 if one has been created as described above. The entry "Show DM Cursor in Images" activates a second play cursor 15b in the PR 29. A first play cursor 15a exists on the timeline itself. This first play cursor 15a moves clockwise around the perimeter of the PR 29 as the slide show is played back. The larger play cursor 15b which equals the height of the image inside the PR 29 moves across the image in the PR from left to right as the slide show plays back. The length of time that it takes for this play cursor 15b to reach the right edge of each image presented within the PR equals the duration editor time 12 for each of the images.

[064] Figure 11a shows another feature of the PR 29 when used for playing back a slide show. A user can at any time, even during the live playback of a slide show, left-click on the resize button 32 of the PR 29 and resize the PR by dragging in

any direction, as illustrated by the PR 37 in Fig. 11a. When this action is carried out, the image inside the PR 29 is automatically resized to match the new dimensions of the resized PR 37.

[065] The PR 29 can be relocated as well as resized. Referring to Figure 11b, to relocate the PR 29 a user would left-click on the Gray Bar 38 of the PR 29 and drag in any direction. The image inside the PR 29 will be relocated with the PR.

More functions of the Play Rectangle (PR).

[066] There is another category located in the DM Play switches Info Canvas. This category is "Play Rectangle Setup." This category provides additional controls for determining the behavior of a PR during slide show playback. There are three basic types of operations for the PR with a slide show:

- Images are forced to fit the shape of the PR regardless of their dimensions or format. In this case, neither of the entries under the sub-category Play Rectangle Setup 35b of Figure 12, are turned on.
- The PR controls the overall dimension of the images that are played in it, but the images themselves control the format. This mode of operation is activated by turning on "Fit Image to PR". This is found under the sub-category "Play Rectangle Setup" 35b of Figure 12.
- The PR changes both its format and dimensions to match the format and dimensions of each image playing within it. This mode of operation is activated by turning on "Fit PR to Image". This is also found under the sub-category "Play Rectangle Setup" 35b of Figure 12.

A. Images force fit to the PR.

[067] When neither the Fit Image to PR nor the Fit PR to Image entries are turned on, the PR is in full control of the geometry of all images playing in a slide shown when the PR is activated. Activating the PR means turning on a PR switch. When this occurs the PR becomes visible in Blackspace and any images playing in the slide show are made to completely fill the inside space of the PR. This means that

if the PR is resized, as shown in Figure 11c, by dragging its resize button 32 in any direction, the current image playing back in the slide show and all subsequent images will have their geometry changed to match the size and shape of the PR. This PR operation has the advantage of permitting users to create slide shows using any size images where all of the images are made to conform to one shape and one set of dimensions, namely, that of the PR 29. The disadvantage of this method is all images, including landscape, portrait and square formats are resized to fit into a single shape, that of the PR. As illustrated in Figure 11c, if, for example, landscape is chosen for the shape of the PR 29, any portrait image will look skewed when it plays inside the PR, because it will be resized to match the landscape format of the PR.

Fit Image to PR.

[068] If the new image playing back in the slide show has got an aspect ratio of portrait and the PR is landscape, it flips the PR's sides between its long side and its short side. To look at this more closely, the PR is made to resemble a landscape format. This means that its width is greater than its height. Now an image that is of a portrait format is made to play in this PR. What happens is that the width and height of the PR are swapped to become the dimensions for a portrait format and these dimensions are applied to the image playing in the PR. The consequence of this is that a user has control of the displayed size for both formats. The user sets the shape of a PR, which then determines the size of all landscape, portrait and square images in the slide show. Square images are explained under the section below entitled: "Square %."

[069] Note: Using both the Slide Show VDACC 11 and the PR 29 onscreen can be very helpful. The PR lets the user control the size and location of the images playing back in the slide show, while the Slide Show VDACC gives the user an overview so the user can see what all of the images are in the slide show, particularly what the previous and next images are with respect to the currently playing image.

Fit PR to Image.

[070] Figure 13 shows the entry Fit PR to Image selected in the category Play Rectangle Setup 35b in the Info Canvas for the DM Play switch 34. Fit PR to Image

means that each image that plays back in a PR determines the dimensions of that PR. As the slide show plays back, each image that is called up to play in the slide show causes the PR to change to match the dimensions of this image in its native state as it is stored on a hard drive or other such storage medium. Let's say that image 36 is a portrait image with dimensions of 4 inches wide by 6 inches high. When this image plays back in a PR, the dimensions of that PR will be automatically set to match the dimensions of this image, namely, 4 inches by 6 inches.

[071] When the next image in the slide show starts to play, the PR will immediately change to match the dimensions of this image. The dimensions that are matched by the PR are the dimensions of the actual picture file.

[072] Returning to the function Fit Image to PR, there are other operations that can be applied to images in a PR when the PR is in the "Fit Image to PR" mode. These are found under the sub-category entitled: "Auto PR options" located in the DM Play switches Info Canvas under the category "Play Rectangle Setup" 35b. Note: The conditions that are to be described are only operational when Fit Image to PR is on. Note: when Fit PR to Image is activated, none of these features matter, as the PR is controlled by the dimensions of the images.

[073] Additional functions associated with Fit Image to PR are as follows:

1. Auto PR retains center point.

[074] Referring to Figure 14, what is shown here is the condition when the entry Auto PR retains center point is *off*. This is the default for the Slide Show software. With this option off all of the images that play back in a PR 37 are aligned by their upper left corners to the upper left corner of the PR. Landscape 38 and portrait 39 images change the dimension of the PR 37 to match the dimension of these native saved picture formats (the dimensions that the pictures have as raw picture files stored on a hard drive, CD or other storage medium). But, each of these images is aligned to the previous image by matching the coordinates of its upper left corner to the coordinates of the upper left corner of the PR 37. In other words, in this mode, the upper left corner of the PR is fixed, although the rest of it will change to match the overall geometry of each new image played in it.

[075] When the entry Auto PR retains center point 40 is activated, as shown in Figure 15, the images playing back in the slide show align their center points to the center point 41 of the PR 29. Landscape images 38 and portrait images 39 are aligned to the center point of the PR 41. Note: all of the images in the slide show can be of a different size. It doesn't matter. Whatever size they are, the PR's dimensions will be automatically adjusted to match them when they become the currently playing image in the slide show.

2. Square %.

[076] Referring to Figure 16, "% square 95" 42 is a user-defined control to determine what shall define a "square" image. Ideally a square image is an image that has four equal sides. The Info Canvas entry 42 enables a user to change this classical definition of a square to something that is not an ideal square. The number at the end of entry 42 can be changed to anything from 0 to 100. Zero would mean that any object would be a square and 100 would mean that only an object that had four perfectly equal sides would be a square. Typing a 95 for this parameter means that any object whose sides are within 95% of the length of each other would be considered to be a square. This feature exists because many photos that appear to be square do not have perfectly matching sides. The PR is designed to change its shape to landscape, portrait and square overall shapes.

3. % of landscape width.

[077] Referring to Figure 17, "% of landscape width" 43 is the last setting for determining what a square is in a slide show. This function exists because if a square image matches the length of each of its sides to a distance that equals the length 44 of the PR 29, square images in the slide show will be quite large, certainly they will feel larger than either the landscape or portrait images.

[078] On the contrary, if a square image sets the length of each of its sides to a distance that equals the height of the PR, square images in the slide show will appear much smaller than the other images. The solution is to enable users to determine what the length of all square images will be in their slide show when the user has activated the "Fit Image to PR" mode. Using a factor of 60% for the "% of landscape

width" 43 means that the length of each side of a square will equal 60% of the length of the PR's landscape length. Notice that the entry "% of landscape width" 43 is followed by a numeral. In Figure 17 this is the number 60. Distance A in Figure 17 is approximately 60% of distance B.

[079] In Figure 17, a square 45 resulting from this calculation is shown. To change this percentage, a user need only place a text cursor next to 60 and type a new number.

[080] A description of the GUI procedure is now described.

User drawn blue arrow which intersects objects onscreen

Intersections are determined by "collisions" in the drawing space.

[081] The software asks the drawing surface for a list of things that collide in a drawing sense with the objects that it's inquiring about. Then it gets a list of objects back from the drawing surface. This list includes the things that collide with the object in question or collide with a certain point on the drawing surface. This is the method for working out what a drawn arrow is intersecting. This same method is used for anything that involves making decisions based on the fact that two things intersect each other.

[082] The software analyzes all objects which overlap each other in any way on the drawing surface to provide a collision list. Objects which are hidden do not collide but objects which are visible but clipped into a VDACC do collide but can be mapped out of the collision list if required. This normally is the case.

[083] The collision list contains nothing which is set to be hidden. There are things that come back from the basic drawing software that are not visible because they are clipped. For instance, if an object or objects are placed into a VDACC and then these objects are scrolled up so that they are above the top edge of the VDACC (they are above the visible area of the VDACC), then these objects are no longer visible. This is because they are automatically hidden when they scroll outside any perimeter of a VDACC into which they are clipped.

[084] If a user clicks in this area with the mouse or its equivalent, this mouse click will not collide with these objects.

[085] The drawing software decides if an object is completely hidden or partially hidden by virtue of the fact that it is clipped into a VDACC, and if this is the case, this object is removed from the list of colliding objects. The point here is that collisions must not take place with objects that the user can't see. Otherwise, the user will not be able to control what the user is doing in Blackspace.

[086] The reason this work needs to be done is because the collision list includes these objects that are hidden to the user. So these objects need to be mapped out of the collision list.

[087] The list of objects which collide with the arrow is sorted according to the nearest point in the list of points in the arrow's tail.

[088] When the arrow is drawn by the drawing surface, the arrow is drawn using a list of points that make up the line that becomes the tail of the arrow. The way to sort the objects that are colliding into the right order is by seeing what the nearest point in the arrow tail is.

[089] The list of colliding objects is a list in no particular order. Collisions are defined as objects that graphically collide with the arrow tail. This list is asked for by the software manager of the drawing surface. The drawing software is software that contains the lowest level of drawing objects.

[090] At this point the list of collisions is stored in the arrow object itself. So the arrow object keeps a list of all the objects with which it collides. Furthermore, the arrow object keeps two additional lists: (1) a source list – the objects that its tail (arrow shaft) collides with, and (2) a target list – the objects that its head or tip collides with.

Analyze where the arrow starts and what it is pointing at

- The arrow provides a collision region at the tip of it's tail and another extending a short distance in front of its head.

[091] A region is designated at the tip and tail of an arrow that enables a user to draw that arrow such that it doesn't have to actually intersect an object in order to collide with it. If an object is within this collision region, it is determined that the arrow collides with it. A default for this collision region is approximately 20-30

pixels. This is a user definable parameter which can be changed by making a selection in the Info Canvas for the arrow being drawn.

[092] Regarding the shaft of the arrow (the drawn line of the arrow) intersections are made just with the line itself. Although a collision region could extend along the length of a drawn arrow's shaft, this would in general decrease the accuracy of the drawing of an arrow for the purpose of selecting items with its shaft. Therefore, there is no collision region along the shaft of a drawn arrow. This shaft (line) must itself intersect an object to create a valid collision with it.

[093] This is important when a user draws an arrow such that it intersects only certain items in a list of graphic objects, e.g., photos, in a VDACC. What is desired here is that what is visually presented to the user matches what actually happens with regards to collisions with the arrow's shaft. So, if the arrow shaft intersects the tiniest part of a letter that is part of the name of a photo, for instance, that photo will be selected. And if the shaft just misses the letter then this entry (this photo) will not be selected. In the case of a slide show, this means that this photo will not be added to the list of images that will play back in the slide show.

[094] The regions at the tip of the tail and the head of an arrow are there to enable a user to more easily select items right where the arrow starts to be drawn and right where the arrow ends. This collision area is valuable here.

- These regions obtain their own collision list and determine the "prime source" at the tail and the "target" at the head.

[095] As previously mentioned, there are two collision lists maintained in an arrow object when it is first drawn: (1) a tail ("prime source" collision list, and (2) a head (tip) "target" collision list. Both of these lists are part of the definition of the arrow object.

- The prime source and the target are determined as the objects with the highest z-level in these two collision lists.

[096] If the target collision list for an arrow contains more than one object, then one of these objects must be chosen to be the target. The object that is chosen is the object that has the highest layer level, what is referred to as the highest Z-level.

[097] For example, let's say an arrow is drawn so that it points to an object that is in a VDACC. In this case, the target would contain the object that is intersected by the arrow and the VDACC itself. Any object sitting in (on) a VDACC has a higher Z-level than the VDACC, so the object would be selected as the target for the arrow. If the arrow tip is on the VDACC and does not intersect any object on the VDACC, then the VDACC will become the target for the arrow. If an arrow is drawn to a group of objects that are on top of each other, the arrow selects that object in this group that has the highest Z-level and this becomes its target.

Remove invalid objects from the list of intersected objects

[098] Below are examples of things which are invalid objects.

Invalid objects:

- VDACCS which contain the prime source or the target.

[099] The way VDACCs work is they enable arrows to be drawn across their perimeter lines without selecting them. Also an arrow can be drawn across the perimeter lines of VDACCs to intersect objects at the tail, and not select the VDACCs themselves. This is a necessary operation for the blue arrow. Users can draw the shaft of the blue arrow to intersect any one or more picture files in a VDACC picture file list and then draw the arrow across a perimeter line of the VDACC and point it to Blackspace. The blue arrow only selects what it intersects, except for the VDACC, which it does not select.

- Parts of the arrow or the original line used to draw the arrow.
- VDACC which contains any of the intersected objects.

[0100] What is referred to as the Picture File VDACC, contains the names of all of the slide show images that may be selected by the blue arrow to create the slide show. The blue arrow is used to insert new pictures into the slide show. This method involves drawing the shaft of the arrow across the perimeter of the Picture File VDACC. When this is done, the Picture File VDACC is not selected by the arrow.

Create an "incomplete" arrow logic using the passed list of objects

[0101] When an arrow whose color matches a know arrow logic is drawn, it has what is referred to as an incomplete arrow logic. When such an arrow is drawn

between one or more objects that are considered valid for that arrow logic, the head of the arrow turns white (any graphic can be used here, e.g., a blinking arrow, flashed arrowhead, etc.). At this point in time this arrow has an incomplete arrow logic. Then when a user left-clicks (or its equivalent) on this white arrowhead, the arrow logic for this arrow becomes complete.

[0102] Regarding the term "passed list of objects", this list is passed from the arrow object to the arrow logic. At this point the arrow logic is saved in memory. The arrow object has collected data from its source and target collision lists. The arrow logic determines which objects are invalid and which are valid and then passes these lists to the arrow logic.

[0103] The arrow logic is a purely software object which exists in memory. Arrow logics are created as temporary files in memory and are used to effect the operation the user is trying to create by drawing of an arrow between two or more objects on the drawing surface. When these logics are used the arrow is discarded and the logic remains in effect in the system software until the original link or links that were established between one or more objects, by the drawing of this arrow, are removed.

[0104] The arrow logic defines operations and it contains a list of objects on which those operations are performed. An arrow logic is a list of sources and targets and the logic has a type which is worked out from the color that was used to draw the arrow and then when things happen on source objects they communicate with the arrow logic which they know they belong to and call methods on the target objects via the arrow logic.

[0105] The "type" of an arrow logic is the action or function that will be carried out when the logic is complete. There are many different types of arrow logics and each logic can be assigned to a different color so a user can easily tell them apart. Let's say for instance, a user draws a red arrow and that this color assigned a "control logic" to the drawn arrow. This control logic can be stated as a sentence as follows: "what the arrow is drawn from (what objects the arrow intersects at its source) controls what the arrow points to (the objects that are arrow's target)."

[0106] The "things" that happen as a result completing an arrow logic are dependent upon the type of arrow logic completed. For instance, if a "control arrow logic" were used between a source fader and two target faders, the "thing" that happens is that as the cap of the source fader is moved, the fader caps on the target faders will move with it, because they are controlled by it.

[0107] In the case of a slide show, the blue arrow has a different logic. This arrow logic takes the items that are collided by the drawing of the arrow's tail and causes the following things to happen when the tip of this blue arrow is drawn to point to blank Blackspace.

- Each of the intersected items are placed into their own small VDACC of a predetermined size, which can be user-defined and altered.
- Each of the small VDACCs are placed into one larger VDACC where the smaller VDACCs are arranged in horizontal rows of 8 small VDACCs each.
- Each of these source items is assigned a start time and a duration time.
- Each start time for each item is delayed later than the previous item by a time that equals the duration of the previous item.
- All of the source items can be played back in sequential order whereas the VDACC that contains each of these items is not played back, only the item itself.

[0108] The blue arrow logic is a sequential logic. It's not limited to images, like photos. It can also be used to play back graphic devices, like faders, switches, knobs, joysticks and the like in sequential order. It can also be used to do the same with text objects, etc.

- An arrow logic linker contains a list of its source controls and a target control.

For example, if an arrow is drawn through a group of photos, the arrow logic linker knows that a list of photos has been intersected as a source, and if the arrow is pointing to Blackspace, there is no target control.

- The arrow logic provides a vehicle for objects in the arrow logic to communicate with each other. This is not required in this case.

An example of this is what has already been cited. It would the drawing of an arrow from one fader to another. This creates an arrow logic which transmits value changes from one fader to another fader.

[0109] In the case of creating a slide show, this is not relevant. These objects are not being asked to send messages to each other. So the drawing of a blue arrow is not requesting the images selected at the arrow source to send messages to Blackspace, which is what the tip of the arrow selects when this type of arrow is first drawn.

- A flag is set which permits the arrow logic to be discarded once it has been processed.

[0110] Once the arrow logic is processed it's no longer needed. This is after the white arrowhead has been touched and the logic has effected its operations, performed its assigned tasks.

[0111] On the processing of an arrow logic, the arrow logic has in part taken all of the images that were created by the result of the arrow being drawn to intersect text that represents these images, then created a small VDACC for each of these images and then placed each image into one of these VDACCs, and then created a Drawmation session, etc. until its finished. Then there's no further need for that arrow logic. For more information on a Drawmation session and Drawmation, in general, see simultaneously filed U.S. patent application entitled "System and Method for Recording and Replaying Property Changes on Graphic Elements in a Computer Environment", which is specifically incorporated by reference herein.).

[0112] The processing of the arrow logic takes place after the white arrowhead of the drawn arrow that initiates this logic has been touched (left-clicked on or its equivalent). All of the above described work which is done by the arrow logic is a result of touching the white arrowhead on the drawn arrow. In summary, when an arrow logic has performed its assigned tasks it is discarded unless it is required to be retained as a functional link.

[0113] The result of the above described arrow logic performing its assigned tasks is a Drawmation session with some specific behavior attached to it. With

regards to a slide show, the arrow logic creates the sequential order of the slide show elements (the source objects for the originally drawn blue arrow) and its job is done.

- A flag is also set to indicate that the drawn arrow can be discarded once the arrow logic has been processed.

The discarding of the drawn arrow results in the arrow disappearing once its white arrow head is left-clicked on by a user.

- In this case, it is valid for the arrow logic to have no target if it was drawn pointing to raw Blackspace.

[0114] Slide shows are created using a special arrow logic that is not a slide show logic. A slide show is just one use of this arrow logic. The arrow logic operates by drawing the shaft of an arrow to intersect various media and then the target of this arrow is Blackspace itself. In fact, the arrow head must point to blank Blackspace in order for this logic to be valid.

[0115] There is an exception to this, in that once a slide show is created you can draw the same blue arrow to the Slide Show Container VDACC to insert more slides.

[0116] This logic is assigned as a default to the color blue. The assignment of colors to arrows is a user controllable function. See pending U.S. patent application serial no. 09/880,397.

[0117] The logic that is used to create slide shows reads as follows: "The objects that the arrow is drawn from (that the arrow either intersects and/or encircles) are each placed into their own VDACC, assigned a default onscreen time, and then assigned a sequential order which is the order that they are played back. This is the order that they were intersected or selected by the drawing of a blue arrow (or another colored arrow that has this logic assigned to it). "

[0118] One might think of this particular arrow logic as a "sequential" logic. All types of media can be intersected or encircled with this arrow and caused to play in sequential order. These media include: pictures (photos), text, recognized graphic objects, free drawn lines, graphic devices (e.g., faders, switches and knobs and joysticks), video, animation (like Drawmation), documents, glued conglomerates (glued groups of items), VDACCs with or without objects within them, and the like.

[0119] This sequential arrow logic is not limited to picture files, e.g., .png, .jpeg, .gif, .bmp, etc. Graphic objects, like stars, folders, circles, etc., can be sequentially presented with this blue arrow logic. Any number of text objects typed in Blackspace or in a VDACC can be presented sequentially with this arrow logic.

[0120] If the user clicks on the arrow head, the arrow logic data is passed to the slide show software. Clicking on the head of a blue arrow (where the arrow head has turned white) is the last step in causing the items intersected by this drawn arrow to be sequentially placed into VDACCs ready to be played back in their sequential order.

[0121] A description of a process to a slide show from an arrow logic is now described.

[0122] 1. Test that the arrow logic is of the correct type. The slide show software makes this test. When an arrow is drawn the color of the arrow is translated into an arrow type, which is performed before this step. By this time color is not part of the equation any more.

[0123] 2. Create a small VDACC and make enough copies of this VDACC to hold all the items in the arrow logic source list. This step creates a VDACC for each item that exists in the arrow logic's source list, for the arrow that was drawn in Blackspace.

[0124] 3. Examine each object in the source list. This source list is from the arrow logic. The source list is generated by the object that the arrow intersects with its shaft.

[0125] If the object is a text object representing a disk file, then send a request to the file management system to create an object from the file. This is most appropriate for files representing pictures. A picture file (photo file) can be represented as a text file in a VDACC for instance. An arrow can be drawn to intersect various text files in this VDACC. These text files can represent image files stored somewhere, e.g., on a hard drive, CD or other storage medium. A flag is set to say that the slide show software is waiting for content to be returned. In other words, a request is sent out to get a picture (photo) from a file name on the source list. The picture is returned by the

file system to the GUI software. When this object is returned by the file management system it follows the procedure below.

[0126] The following procedure is performed when the graphic object is returned.

- If it's any other text object, it is resized so that it has the appearance of being approximately 12 point in size.
- If it's any other kind of graphic object it is resized to the size of the small VDACC.
- Each object is processed like this is placed into one of the small VDACCs. At the same time, it is locked to the surface of the VDACC. Each of these VDACC is now treated as a slide.

[0127] 4. As each object is placed into its small VDACC, the VDACC is assigned a start time and a duration based on settings stored in the slide show object. The slide show object is the software construct, the slide show management software, which is managing this process. This slide show management software has a list of the start times, end times and duration times for each item that has been placed into a small VDACC.

[0128] The user controls these settings via entries in the slide show Info Canvas, which is part of the DM play switch info canvas. The start time stored is cumulative based on the calculated end time of the previous slide. In other words, each new slide's start time is at a time that equals all of the previous slides before it added together. As each object is assigned a time, a text object is created which is "connected" to the slide. The "slide" is a small VDACC with an image in it. The text object that is added shows the duration time for the slide. This text object is generally placed above the slide, but it could be placed anywhere. When the user edits this text object, the duration of the slide is altered accordingly. See "Changing the onscreen time of one or more slide show images" described above.

[0129] 5. When all the content has been acquired as above, the slides are ordered and placed into the slide show VDACC. The size and position of this slide show VDACC are calculated to show horizontal rows of eight slides, which is the default for the software. However, the slides could be organized in many different

arrangements. This arrangement is then shown to the user. The word "content" here refers to all of the individual items that comprise the slide show, including the requests that have been sent to the file system to get pictures from it.

[0130] 6. Create a Drawmation session.

- A request is sent to the Drawmation manager to create a session long enough to include all the slides. This session generally equals the length of the sum of all the duration times for all of the slides in the session.
- A Drawmation event to show and hide each slide is sent to the Drawmation manager. Each slide is set to have a "hide" event which is 1ms later than the "show" event for the next slide. This is so that the user perceives no gaps between adjacent slides. There is an event assigned to each of the small VDACCs in the slide show to have the item assigned to it to show (be visible) for the duration of time that is set by its text object and then hide it for the rest of the duration of the slide show. There are two events here, a show event and a hide event that are created for each slide. The hide event for each slide equals the start time of that slide plus its duration.

[0131] 7. Although the slide show management software keeps notes of the times for various features, in order that they can be edited, saved, etc., it is now the Drawmation system which is the holder of the authoritative information.

[0132] In other words, what happens when a user activates the Drawmation play switch to play the slide show is entirely under the control of the Drawmation software, not the slide show management software.

[0133] Turning now to Figure 18, a flow chart for playing a slide show is shown. All real time behavior is controlled by the Drawmation system. Real time behavior means the real time manipulation of graphics on the drawing surface. Drawmation works by sending out event messages, and the timing of these messages is the responsibility of Drawmation, not the slide show software.

[0134] When a user does anything graphically in Blackspace (on the computer screen) this is recorded by maintaining a record of all the event messages associated

with those actions. These event messages are managed by the Drawmation system. In the case of a slide show the event messages are created partially by the arrow logic which creates the sequential order of the items and places them in VDACCs, combined with any inserts or edits that a user would add to the slide show subsequent to its initial creation by the drawing of the arrow. The slide show provides its own features by intercepting messages coming from the Drawmation system to the GUI. Most of these are "show" and "hide" messages.

[0135] Intercepting messages is a basis for the slide show software. What the slide show software does is intercept messages that it understands. What it does in Drawmation is to send a set of show and hide messages and have those replayed. When Drawmation normally sends a show message, that message has a defined behavior. It causes an object to become visible and that's the point.

[0136] But the slide show object requires a more complex behavior. So the slide show software intercepts this message and doesn't just show the item, the slide show software modifies the item according to features controlled by the slide show software. This is what this flow chart shows.

[0137] *Is a message received?* This is a message from Drawmation. Drawmation is sending the message to the GUI. The reason the Drawmation is sending it is because that message's time has arrived. An example of this would be this. There is an image in one of the small VDACCs in the slide show VDACC. The start time of this image equals the current time of the playback. Because it's start time equals the current time, a message is sent to the GUI to say show this image. The image that just finished playing, has a message that is sent to the GUI that says "hide me."

[0138] Drawmation doesn't know anything about slide shows. It's just sending out messages. Drawmation is the system for having the system play things in real time. It does this by sending messages to the GUI to tell the GUI what to do. When a slide show is created, a set of messages are artificially inserted which are simplified to the extent that the messages are just a show message and a hide message. When these messages are played back, the slide show recognized that they are either a show or a hide message for one of its members and stops that message from going straight

to the object and applies various behaviors to that message through the slide show software.

[0139] The slide show software acts as a monitor on the Drawmation messages and invokes special behavior if the message fulfills certain criteria.

[0140] When a user is playing back a Drawmation session, it contains a list of events which it is sending out when the right time arrives for that event to happen. Creating a slide show artificially creates a series of events for the objects that the slide show is managing.

[0141] When those events are played back by Drawmation, the slide show software recognizes that that event is one that it created. This causes the slide show behavior to be invoked for that event, as opposed to passing it straight to the GUI.

[0142] Then, *is message "modify"*? What kind of message is this? Is this a "modify" message? There are various message types in Drawmation. This is one of the message types.

[0143] Then, i*s message aimed at a member of the slide show?* In other words, is the message aimed at one of the graphical elements that is under the control of the slide show software? If the answer to any of these first three questions is "no", the software aborts.

[0144] Then *find that slide.* The software finds the image that is currently being played.

[0145] This means find the small VDACC with a certain start time and duration time. The message is not recognized by virtue of the time. It is recognized by virtue of the ID for that slide. The slide show software has a list of IDs for the objects that make up the slide show. So it recognizes that this message is being sent to the ID of one of its objects (which could contain an image, a text object, a graphic device or other type of graphic object).

[0146] Every object in the system has an ID. Not just graphic objects, but also system objects and software objects have unique ID s. So the slide show can know which objects belong to it just by looking at the ID for each of those objects. At this point the slide show software doesn't need to know about time.

[0147] The Drawmation sends a message to the GUI saying: "Show an object with this ID." There are no times in this message. The slide show software looks at the ID of an object and if this ID is in the slide show software list, then it says "this is one of mine."

[0148] The ID of the object in the message will identify one of the slides. The slide show software "finds that slide" using the ID in its list.

[0149] So the slide show software knows that there is a message. It knows that the message is aimed at a member of its slide show, which is a slide. And it knows that in that message is the ID so it can find the slide based on the ID in the message.

[0150] Then, i*s message "show"?* One of the modify messages is "modify visibility". This is saying, is this a show message, namely, make visible?

[0151] If no, then *is message "hide"?* If no, the process is done.

[0152] If yes, then *is full screen mode on?* If yes, then *rescale slide contents to fit the screen*, and the process is done.

[0153] If no, then *is "show image in VDACC"* ON? If yes, then *set duration text green.*

[0154] When an image in the Slide Show VDACC is on (it is the image that is currently playing), the duration text (what is referred to as the onscreen time parameter) for that image turns green. Normally this parameter would be white. The onscreen time parameter is the number above the image in the Slide Show VDACC. This parameter determines how long the image will remain onscreen in the slide show. This text turns green to show the user that this is the current slide that is being played in the slide show.

[0155] *Does PR exist and "show in PR mode" on?* The software checks to see if the Play Rectangle exists. Now this doesn't necessarily mean that the PR is itself visible. In fact the PR may be invisible, but its effect will be visible. Its control will be visible.

[0156] When a slide show has been created, turning on a PR switch causes the images being played back in the slide show to play back such that each image is

automatically resized to fill the inside area of a PR. When the "show in PR mode" is turned on, then the PR itself can be turned off.

[0157] There is no question that the PR still exists because the slide show image remain resized according to the size of the PR that was just visible onscreen. The PR still governs the size of each slide show image during playback even though it becomes invisible when the PR switch is turned off.

[0158] Turning the PR switch off is a logical thing to do, as it permits the slide show images to playback as images in Blackspace without anything added around their perimeter. This is, of course, exactly what the PR appears to do when it is visible. It adds the PR timeline around the perimeter of each image in the slide show.

[0159] If no, then *display slide in its VDACC.*

[0160] If the item is already present in its small VDACC, there is nothing to do here. But if is not visible onscreen, the image is resized to fit back into its VDACC.

[0161] *done*

[0162] Turning now to Fig. 19, a flow chart for showing a slide in a Play Rectangle is shown. *Is "fit PR to image" on?* This entry is found in the DM Play switches Info Canvas under the category "Slide Show." Fit PR to image means that the dimensions of the Play Rectangle timeline are changed to match the dimensions of each successive image playing back in the slide show. In other words, the Play Rectangle changes its dimensions to perfectly match the size and shape of each image as it is stored on a hard drive or other such storage device where it is being recalled from for the slide show.

[0163] *Is image square?* If yes, then *set PR to be square.*

[0164] *Is PR square?* If yes, *restore previous PR dimension,* then ask *do PR and image have the same aspect ratio?*

[0165] If when playing a slide show one image is square and then the next image is not square, the square shape of the PR needs to be undone.

[0166] Referring again to *is PR square?* If no, then *do PR and image have the same aspect ratio?*

[0167] If yes, then *resize object to fit PR dimensions* and the process is done. If no, then *switch PR between landscape and portrait.* Then *resize object to fit PR dimensions.*

[0168] Referring back to the first question: *Is "fit PR to image" on?*

[0169] If no, then *is* "fit image to PR" on? If yes, then *is image a picture?* If yes, then *set PR size to picture size* and then *resize object to fit PR dimensions.*

[0170] If no, then *is image a text object?*

[0171] If yes, then *set text to original font size (remove all scaling).* Then *set PR size to size of text* and then *resize object to fit PR dimensions.* If no, then *resize object to fit PR dimensions.*

[0172] Fit image to PR is also found in the DM Play switch's Info Canvas under the category Slide Show. Fit image to PR means that each image in the slide show has its geometry changed to match the dimensions of the Play Rectangle. These dimensions can be changed by the user at any time, even while the slide show is playing back.

[0173] To accomplish this the user left-clicks on the resize button of the Play Rectangle, located in its lower right corner, and drags in any direction. As the Play Rectangle's dimensions are changed, the slide show image shown inside the Play Rectangle is changed to match the new dimensions. Furthermore, then this change occurs for the image in the Play Rectangle, this change is automatically applied to each and every subsequent image which plays back in the slide show, until the dimensions of the Play Rectangle are changed again. In each case, the new dimensions for the Play Rectangle will govern the dimensions of every image that plays back in the slide show from that moment on.

[0174] Regarding the step "*set text to original font size (remove all scaling)*", the slide show software is not limited to playing back images (pictures) as slides. It can also play back text typed on the drawing surface. In Blackspace, typed text is itself an object. Each time a user places a text cursor and types text, a text object is created. So multiple text objects can be created and used as a source list by intersecting them with a drawn blue arrow (or its equivalent). Then these text objects will play back in

a slide show just as picture files, e.g., .png, .jpeg, .bmp, .gif, etc., can be played back in a slide show.

[0175] Additionally, recognized objects and graphic devices can be made to play back as slides. The method of their playback is the same as that for images and text objects.

[0176] Turning now to Figure 20, a flow chart for hiding a slide in a slide show is shown.

[0177] Se*t duration editor white.* The duration editor is a small text parameter that appears above each small VDACC which contains each image in the slide show. The default time for this duration editor is 5.000 seconds.

[0178] Then is *"show in vdaccs" on?*

[0179] This is turned on by activating the entry "Show images in VDACCs" which is found in the Info Canvas for the DM Play switch under the category "Slide Show."

[0180] If yes, then *is slide currently in PR*

[0181] If yes, then *does slide contain a text object?*

[0182] If no, then *resize image to the dimensions of the VDACC.* Then *place object back into the VDACC* and the process is done.

[0183] This means the image is resized to fit into a small VDACC which appears in the Slide Show VDACC.

[0184] Referring again to *does slide contain a text object*?

[0185] If yes, then *resize text object so that it has the appearance of 12 point.* Then *resize image to the dimensions of the VDACC* and *place object back into the VDACC* and the process is done.

[0186] Text is resized to have the appearance of 12 point so that text objects of any size, can be read inside the small VDACCs in the Slide Show VDACC. For instance, let's say that a user typed a text object using 48 point Times New Roman. If a font of this size were to be shown inside a small VDACC that is approximately an inch square, this text could not be read. Even if the text were skewed to fit into the small VDACC, it would not likely be easily readable. The solution is to have the text

rescaled to 12 point text which will easily fit into a one inch square VDACC and be readable.

[0187] Referring back to is "show in VDACCs" on? If no, then *hide image*. Then *is slide currently in PR?* The flow chart from this point is described above.

[0188] Turning now to Figure 21, a flow chart for using a Play Rectangle is shown. The user can create a Play Rectangle for use with the slide show. This is done by creating a switch and labeling it "PR". This causes the Drawmation manager to log the switch as one it is interested in.

[0189] The Drawmation Manager keeps a list of timelines and switches which are related to Drawmation. If a user types PR on a switch, the Drawmation recognizes "PR" as something whose functionality it knows about. The Drawmation manager will then store the existence of this switch in its own internal list.

[0190] Then the user would press this PR switch - this event is intercepted by Drawmation Manager

[0191] When the user turns on the PR switch a message is sent out to the Drawmation manager (amongst others) to say that the switch has been pressed. The Drawmation manager acknowledges the existence of this switch.

[0192] Drawmation manager requests the GUI to create a Play Rectangle (a rectangular timeline).

[0193] When the PR is created the Drawmation Manager logs the association between the PR and the switch. In future, each time the switch is pressed, DM causes the PR to be shown or hidden.

[0194] When a PR is created, this event is intercepted by the slide show object, which follows this procedure...

[0195] This slide show object is the slide show manager software object.

[0196] *Slide show receives "PR created" event*

[0197] is "*show image in PR" ON and "show image in VDACC" OFF?*

[0198] If yes, then *hide the Slide Show VDACC.*

[0199] Then is "show image in PR" ON?

[0200] If no, then *tell the PR to notify the slide show object whenever its geometry changes* and the process is done.

[0201] This means that if the slide show has one of its objects in the Play Rectangle (PR) and the dimensions of the PR are changed, then the slide show resizes the object to what it's just been told the new size is. The resizing of the Play Rectangle would be accomplished by a user clicking on the resize button of the Play Rectangle and dragging any direction.

[0202] If yes, then follow the flow chart entitled: Playing a Slide show.

[0203] Turning now to Figure 22, a flow chart for changing the Play Rectangle geometry.

[0204] When the slide show receives notification from a PR that its geometry has changed ...

[0205] *Is there a slide currently being shown in the PR?*

[0206] If yes, then *resize the slide to fit the new geometry* and the process is done.

[0207] If no, then abort the process.

[0208] If a slide show is playing back in a Play Rectangle (PR) and the PR is resized or moved onscreen, then every slide in the slide show from this moment forward is resized and appears in the new location to which the PR was moved.

[0209] Whenever the Play Rectangle's geometry changes which includes being moved, it tells the slide show software and the slide show software will then cause any image which it currently has in that Play Rectangle to also move and resize.

[0210] Every time an event happens which causes a slide to be put into the Play Rectangle then it looks at the current PR dimensions and makes that the size for the image or whatever is playing back at that time in the slide show.

[0211] Every time an event happens which causes the slide show to place an image into a Play Rectangle, then the slide show software asks the Play Rectangle how big it is and resizes the image accordingly.

[0212] The Play Rectangle notifies the slide show object whenever its geometry changes.

[0213] Although specific embodiments of the invention have been described and illustrated, the invention is not to be limited to the specific forms or arrangements of parts so described and illustrated. As an example, the invention may be used to clean and/or grow an oxide layer on an object other than a semiconductor wafer. In addition, the desired reaction may be a reaction other than oxidation using a gas other than ozone. The scope of the invention is to be defined by the claims appended hereto and their equivalents.